Augmenting Artificial Intelligence Creativity:

Event Segmentation and AI planning for Quest Generation

by

Mujahid A. Sanni

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Masters

in

Cognitive Science

Carleton University
Ottawa, Ontario

Abstract

The persistent rise in demand for content in the gaming industry means that programs with the ability to produce content autonomously will save an extraordinary amount of time and cost. In that context, this thesis proposes a novel architecture for procedural narrative generation and implements it as software called Automatic Quest Planning Generator (AQPG).

The architecture combines the computation of Event Segmentation Theory (EST) with Artificial Intelligence (AI) planning methods to automatically generate quests. The idea in this work is that since Event Segmentation generates actions and goals to facilitate planning in humans, a good representation of these types of events either in natural language text or other media should be able to automatically generate planning components for an AI planning system as well.

The system takes in natural language stories and story world entities as input and afterward performs segmentation processes that store the input as events through systematic identification of event boundaries. The system then extracts information from the events based on rules developed in this thesis to generate planning components that are necessary to generate quests. Results from this work found that the AQPG system is capable of generating planning files that can be used for quest generation.

Acknowledgements

First and foremost, I would Like to thank God for giving me the gift of life and strength to see this through. To my family, no dedication can express my gratitude to all of you for your tireless support, especially my dear mother. I pray that God almighty preserves you and grant you a happy life.

I would like to especially thank my supervisor, Dr. Jim Davies for his endless support from the first day I started my program. I am grateful. Thanks to all the committee members for their suggestions and for also agreeing to be on my committee.

I would also like to thank all the friends in the Science of Imagination Lab and in the Department of Cognitive Science. You all made the journey bearable. I appreciate you all.

List of Tables

List of Figures

Chapter 1

Introduction

Previous academic studies in cognitive science have shown evidence that narrative construction is how we make sense of everyday events (Baddeley, 2003; Zacks et al., 2007). More evidence indicates that an Event Segmentation process is at the focal point of narrative construction in humans (Zacks et al., 2007; Baldassano et al., 2017). Consequently, the Event Segmentation Theory has been investigated in wide breadth in terms of understanding the mental process involved, but there are hardly any studies that have tried to explore this discovery in terms of generating narratives as media content.

The demand for media contents keeps rising, especially in the video game industry. This means that the concerned industry is faced with the responsibility of the increased cost of paying artists and programmers to create more game content. In this context, research and studies geared towards the automatic generation of content have attracted significant attention due to this high demand of such content by game players. Programs that can produce quality content on their own save a remarkable amount of time and cost.

1.1 Procedural Content Generation (PCG)

Procedural content generation in games refers to a technique for automatically creating content for games through algorithmic means. Levels, maps, stories and characters are all types of content created with PCG in games (Shaker, Togelius & Nelson, 2015). The gaming industry employs thousands of artists and programmers in an attempt to meet their customers' demand for dynamic and interesting content. However, small game companies cannot afford such manpower.

The alternative to PCG for narrative generation involves narratives written manually and fed to the system during the design stage. With PCG, computer programs can automatically create narrative content that can adapt to users' preferences. Consequently, PCG is a promising area for progress in modern games. Many approaches to procedural content generation have been introduced over the years, such as computational linguistics (Elson & McKeown, 2010), narrative theory (Herman, 2007), and multi-agent architectures (Osborn, 2002).

One of the current topics in big game companies is that present methods of PCG show a lack of dynamically generated contents that are sufficient enough to keep game players entertained. The results of the content generation have become too generic. As Togelius et al. (2013) rightly pointed out while discussing the challenges in procedural content generation:

> "Generated content generally looks generic. For example, looking at the dungeons generated for a roguelike game such as those in the Diablo series, you easily get the sense that this is just a bunch of building blocks hastily thrown together with little finesse."

In most present PCG games, the objects, worlds, scenes and game levels tend to feel the same as the players progress in the adventures.

PCG is primarily a very crucial topic for small independent game development studios due to low budgets, where PCG can generate content for less effort and human resources (Korn & Blatz, 2017). Aside from the cost, both big game development studios and small game development studios need solutions that will reduce development time. As noted by Hendrix, Meijer, Van der Velden and Losup (2013), the massively multiplayer online role-playing game (MMORPG) World of Warcraft (2004) was developed during a period of almost five years without using PCG. It was comprised in 2008 of over 1,400 locations, 30,000 items, 5,300 creatures, 7,600 quests, and 2 million texts.

That being so, more research on new methods of automating content generation such as this is necessary to move the gaming industry forward.

1.2. Narratives

Storytelling is one of the oldest known human activities (Wiessner, 2014). Riedl (2016) said "Storytelling is an important part of how we, as humans, communicate, entertain, and teach each other." The importance of storytelling to humans cannot be overemphasized. It serves as a means of education, culture preservation, entertainment, etc. As a source of entertainment, stories are turned into books, movies, etc. For culture preservation, different cultures keep records of their history and customs in the form of stories. In education, aside from learning history, storytelling has been used as a strategy

to improve literacy (Miller & Pennycuff, 2008). The ability to craft, tell and understand stories has long been held as a hallmark of human intelligence (Winston, 2011). These abilities are referred to as Narrative Intelligence. Li, Lee-Urban, Appling and Reidl (2012) define Narrative Intelligence as the ability of an entity to organize and explain its experience in narrative terms, comprehend and make inferences, and produce responses to narratives. It is interesting to investigate computer narrative generation using a method that was inspired by a human psychological process.

1.3 Quests

Ammanabrolu et al. (2019) explains that quests consist of activities that are partially ordered, and the goal of the player is to reach the end of the quest by successfully engaging in those activities. In most adventure games, a player is given a quest to complete based on a storyline. Once the player completes the quest, they can advance to the next level. Quest generation is a type of narrative intelligence and involves creativity just as any other type of narrative. Narrative intelligence is required in designing quests because coherence must be maintained as the quest progresses towards its goal. This is one of the reasons why planning is also suitable for quest generation. Other reasons include balancing character believability and goals (Young, Ware, Casell & Robertson, 2013). Chapter two has more discussion on coherence and character believability.

1.4 AI Planning

Planning is a crucial field of research in Artificial intelligence. Over the years, it has been used to solve a wide range of real-world problems. For example, mission planning in the military (McKendrick, 2017), autonomous car controls (Caporale et al., 2018), e-learning

planning (Garrido, Morales & Serina, 2012) etc. It also makes sense to see story generation as a planning problem because the way AI planning algorithms work is to search for the best action sequences that fulfill a goal (Breault, Ouellet & Davies, 2021). In a story world, stories unfold in a sequence of events. If we apply that to a game world, we can say that quests unfold in a sequence or partial order of events that must be completed to achieve a goal. A solution to a planning problem is a plan, which contains a sequence of actions. A plan is successful only if it reaches the intended goal. Domain Knowledge is usually required in planning systems, i.e., the planner must know every possible occurrence in the world of the problem (the game world in this case). Any planning problem is based on these three concepts (Breault, Ouellet & Davies, 2021).

- Initial state: A set of statements or literals that describe all the facts that exist at the beginning of the problem/world.
- Goals: A description of what we intend to achieve with a plan. These take a form statements that must come true in the world.
- Domain: A set of available actions and rules to achieve the described goals.

Given these three concepts, AI planning has proven to be an effective way of representing narratives (Young, Ware, Cassell, and Robertson, 2013) by using first-order (predicate) logic. A planning problem is composed of a specification of an initial state describing a world in its current or beginning state. This can be represented as quest worlds created by authors. The quests can be represented as different goals in the planning system that must be achieved by players. The use of actions that achieve a set of goals in planning is a good representation of quest problems as well as other narratives. Planning also

provides a wider space of stories to explore than other methods such as scripting and story grammars (Riedl and Young, 2013) since actions can be reused and the planner has the freedom to add actions.

However, planning for narrative generation is not perfect. The restricted first-order logic of representing story worlds and characters means that planners have less human story knowledge (Mueller, 2014) when compared to other methods (e.g., story grammar and scripts).

1.5. Event Segmentation Theory

Event Segmentation (EST) is the process by which humans parse a continuous stream of activity into meaningful events (Zacks, Speer, Swallow, Braver & Reynolds, 2007). Barker and Wright (1954) explained that an event may be very brief (a lightning strike) or very long (the birth of a solar system) but psychological events that are important to study of event segmentation have durations in a human scale, spanning from a few seconds to tens of minutes, e.g., opening a door or driving to a store. Such events have specific structures and involve particular objects, characters, and goals (Speer, Swallow, Braver, Reynolds, & Zacks, 2009).

Humans make sense of daily activities by constructing and storing those activities as events. Zacks et al., (2007) further explained that sense-making of a current event starts from a representation in working memory, which is known to be actively online but has limited capacity and duration (Baddeley, 2003).

The second popular notion about Event Segmentation is that humans use prediction of coming events to facilitate event segmentation (Baddeley, 2003; Zacks et al., 2007). For event segmentation to be possible, event boundaries need to be constructed to identify when an event ends and when another begins. Event Segmentation proposes that a human's perception of event boundaries is a result of using prediction to form working memory perception (Zacks et al., 2007). That is, the segmentation of events occurs in the mind when the prediction is wrong. If the prediction is right, then no new information is added. Zacks et al. (2007) explains further that:

> "Event models contribute to perception by facilitating predictions regarding what is likely to happen in the immediate future. When relevant dimensions of the ongoing event change, the event model becomes outdated, leading to prediction errors. The system uses those prediction errors as a signal that the model needs to be updated. For example, when watching a clerk bag groceries, one forms a mental model that allows predictions, e.g., the clerk has placed an item in the bag and will now reach for the next item. However, when the last item has been bagged and the clerk is ready to take payment, the old model will generate inaccurate predictions and a new model needs to be established."

The prediction of coming events shows that the mind constructs events from simulation of past experiences during perception of another event. Gärdenfors, Jost, and Warglien (2018) argue that actions and causal reasoning are two of the most crucial aspects of event construction. This means that reasoning about action and its underlying effect is critical in constructing narratives in human minds. A computation of this strategy is particularly useful for narrative generation. It will involve a process of storing events in the memory

and using the stored events to generate actions and reasoning about the effects of actions that lead to a goal.

Therefore, since event segmentation facilitates explicit planning in humans through segmentation of events, I propose in this thesis that a computation based on event segmentation can generate planning files for an AI planning system for quest generation. Traditional AI planning systems require human authors to specify actions and effects based on a story world. Using a computation of ES to automatically generate actions, causal effects and objects in a game world will reduce efforts put into manually authoring game narratives and improve automation of a game generator's creativity.

Chapter 2

Literature Review

This chapter describes some of the past works that are related to this thesis. The first part discusses different approaches of AI planning that have been used for generating narratives. The next discusses the efforts that have been made on investigating Event Segmentation Theory and how they both motivate this work.

2.1. AI Planning and Narrative Generation

Several studies have shown the success of AI planning for narrative generation. Mateas and Stern (2002) and Riedl and Young (2010) extensively discuss how to balance plot and character in AI planning quest generation. Breault, Ouellet and Davies (2021) created an engine for the procedural generation of quests by using an AI planning approach. Most studies (Stern, 2002; Riedl & Young, 2010; Breault, Ouellet & Davies, 2021) agree that of all the factors that contribute to the success of narrative generation, plot-balancing and character believability seem to be a universal focus.

“Character believability is the perception by the audience that the actions performed by characters do not negatively impact the audience’s suspension of disbelief” (Riedl & Young, 2010). The actions of the characters in the narrative should be perceived by the audience as being intentional. Those actions must align with the characters' traits and a set of beliefs. Riedl and Young (2010) also explain that to balance the plot of a narrative, the logical progression must obey the rules of the world in which the narrative occurs. That is, the occurrence of events in the world must be coherent and each event must be related to the eventual outcome of the story.

Some studies focus their approach on addressing character believability in creating sensible narratives. In those studies, the characters are at the center of controlling the narrative. Porteous, Cavazza and Charles (2010) designed a planning algorithm based on the characters' points of view. Cavazza, Charles and Mead (2001) also designed a system that uses the roles of characters to guide the narrative. A more recent variant method of character-based approaches aside from the ones mentioned above have to do with using a simulation of a character's emotions (Griffith, 2018).

The plans in plot-based systems differ from the character-based approach in the sense that a single plan (agent) controls the plot (Cavazza et al, 2001). Magerko, Laird, Assanie Kerfoot, and Stokes (2004) designed a game with this centralized reasoning approach. The game assumes that a single player is the focus and the narrative depends on the actions of that single-player, which leads to a logical progression of events in the game.

To account for balancing plot and character believability, the most recent approaches combine the two methods (character based and plot based) to form a hybrid process. For example, Reidl and Young (2010) designed a framework to maintain the logical causal progression of the plot and character believability in procedural quest generation. Their framework introduces a story planner that also reasons about character believability.

The use of a planning system provides a very reliable way of procedurally generating narratives (Porteous, Cavazza, & Charles, 2010), however, encoding these problems manually by humans can be boring and costly, depending on how complex the story world and the storylines are. This problem is what Cullen and Bryman (1988) referred to

as the Knowledge Acquisition Bottleneck: the extravagant cost of employing humans to convert real-world problems to inputs for an Artificial Intelligence system.

To the best of my knowledge, automatic generation of planning files hasn't been explored for quest generation as of yet, however, I found few studies that have explored the idea for other purposes. Some early works such as Ferguson et al (1996) used mixed-initiative planning to provide users with means of guiding plan construction through interactive ways. The users interact with the planning system through a combination of graphics and natural language display. Other initiatives (Kim et al. 2017) used a form of back and forth user interactivity to automatically generate the plans.

Thue, Schiffel, Arnason, Stefnisson and Steinarsson (2016) designed a plan-based interactive story system that lets authors assign consistent roles to entities throughout a story. The method involves authors describing constraints to guide the assignment of roles to specified entities at different intervals to ensure continuity of generated narratives.

Li, Lee-Urban, Appling and Riedl (2012) described a technique that automatically generates simple events from crowdsourced scripts. These scripts contain schemas that are useful for encoding events in narratives. Li et al. required specific topics and also required specific constraints on those crowd sourced stories. The crowd sourced participants were instructed to (1) Use linear narratives and simple natural language to describe narrations, (2) Use proper nouns for all the characters, and (3) Segment the narratives themselves into sets of events.

Stefnisson and Thue (2018) also created a system called Mimisbrunnur to assist interactive story authors come up with story contents. The authors are required to create story outlines and represent their story in a restricted natural language imposed by the Mimisbrunnur system. The system also requires that the author writes the initial state, actions, preconditions, effects, goals and entity relationships.

Coming up with a storyline is arduous enough for many authors, coupled with writing out the story in a very restricted format and manually encoding elements (actions, conditions, goals, etc.) needed by the system. The proposed Automatic Quest Plan Generator (AQPG) system works differently from the studies described above. The system will encode its planning elements (actions, goals, initial state) through natural language processing of stories and manually labeled entities that the system will take as input. The AQPG system can also use stories that are written for other purposes (e.g., movies) to generate its own actions, conditions, goals, etc., and does not impose any constraint on the author except that the story should be written in English natural language text. The AQPG also does its own segmentation of the stories automatically, unlike in Li et al.'s work, where participants had to segment their stories manually.

A closer work to this thesis in terms of automatically generating planning files is the work of Castillo et al. (2009), where they introduced the use of correct labeling of the Learning Objects Repository (LOR) using specific metadata for automatic generation of problem files for AI planners implemented in e-learning systems. The method required no back and forth user input but it required specific metadata and specific types of databases.

The main contribution of the current work is that it does not only introduce the idea, it also provides a novel architecture for automatically generating the problem and domain files that are needed for an AI planning procedural narrative generation, without the need for a specific database to store the data or require back and forth input from the game designer.

2.2 Event Segmentation Theory (EST)

Even though Event segmentation Theory (EST) has been extensively discussed in cognitive science, Event Segmentation (ES) from text hasn't received a lot of attention. Brain imaging techniques, however, have been used extensively to investigate the theory of ES.

In 1973, Newtson developed a framework for assessing how everyday activities are segmented into events. In his experiment, participants were tasked to watch videos of people performing mundane activities and to press a button when they felt that the activity had ended and another one had begun. Many variants of Newtson's studies have been conducted over the years to investigate ES. One of them was Zacks et al. (2010), during which brain activities of participants were measured while the participants viewed a narrative film. Zacks et al. (2010) observed the time when the activities were segmented and the corresponding brain regions. The result of the experiment supported a proposal made by Newtson (1976) that event boundaries automatically occur when features such as location, time and goals change.

Another study by Zacks et al. (2007) proposed that participants tried to use prediction during ongoing perception for anticipatory actions. The event segmentation is responsible

for predicting the missing cues and simulating the events into a coherent narrative as if they came from perception. This predictive process allows humans to anticipate events before they happen. Cues and patterns of previously experienced events are already imagined before they occur while experiencing a current event that will eventually lead to those imagined experiences.

Further studies on ES over the years investigated whether ES supports the construction and storage of narrative media such as texts, videos and graphics (Zacks et al., 2009, Chen et al., 2012; Kurby & Zacks, 2012) by identifying event boundaries while participants completed a perception task using narrative media that have already been created. However, as of yet, to the best of my knowledge, no known study has tried to direct the knowledge of Event Segmentation to generate creative media.

In general, little has been done to explore how the stored events can computationally generate narrative. Nevertheless, a few studies have tried to come up with a computational model of how ES works to generate narratives. An example of such is Anderson (2015), which proposes a cognitive architecture of narrative generation. Anderson (2015) in his work only explained generating narratives in terms of how it fits into the Soar architecture of cognition, no real work was done on creating media content. In another notable work, León (2016) describes an overall architecture of narrative memory. He extensively covers the use of both episodic memory and semantics to create functions of generating narrative structures. But just like Anderson, (2015), the study did not proceed further to implement those functions in identifying or creating substantial narrative contents. Studies that have at least used computation to identify event boundaries in media contents have focused primarily on visual contents by using event

cameras and sensors to track object movements (Wan et al., 2015; Alonso et al., 2019, Stoffregen et.al, 2019).

The above works of literature and related works clearly show that the approach in this thesis is very distinct from past studies in both narrative generation and computation of Event Segmentation Theory (EST).

Chapter 3

Methodology

In this thesis, I introduce an architecture for automatically generating domain and problem files that can be used by an AI planner to generate quests. The novel architecture involves two levels of segmentation of events written in natural language text. The two levels of segmentation follow a novel event model that I designed. The output of the system is a domain file and problem file, which are generated from extracting elements from the segmented events via algorithms written in this thesis that make use of natural language processing frameworks.

Figure 2 depicts an overview of the system's architecture. The rest of the chapter discusses the system's architecture in more detail.

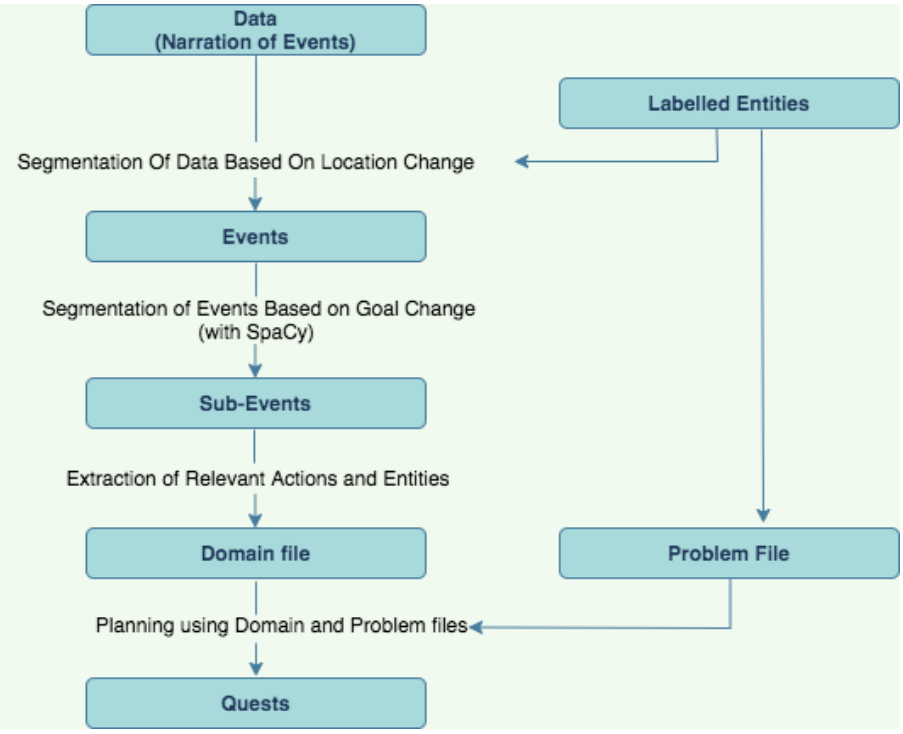

Figure 1. High Level Flow Of The System

3.1 Data

As explained in chapter one, humans use narrative experiences to construct new events. In this work, the equivalent of those narrative experiences are narrations of quest plots. The expected first input of the system is a narration of quest plots retrieved from quest databases online. For this work, I used plot stories of games from fandom.com and Wikipedia. These plots are in natural language text and did not need any fine-tuning or cleaning for the system to make use of them. The reason for the decision to use quest plot narrations is that quest plot narrations have a structure similar to the events described by Zacks et al. but in a more complex form. It would have been easier to use a less complex representation of events, for example using a simple structure of events like the ones described by Zacks et al. (2007), but a system with fewer constraints on inputs depicts more autonomous creativity as it will reduce most of the manual processes involved and also gives designers more freedom to even use data from other sources without needing to create one themselves.

It also makes sense to design the system around plots of role-playing games because it's usually one of the first things game designers finalize before any development of the game takes place. Therefore, the data is readily available and doesn't require any extra time and effort to create.

3.2 Entities Labeling

The second type of data that the system needs are labelled entities. The entities of a story are defined here as all the important elements used in the narration of an event: for example, the locations, characters, items that are important in how series of events unfold.

Named Entity Recognition

Named Entity Recognition (NER) is a very active area of research and an important task in Natural Language Processing (Sahnoun, 2019). Most NLP frameworks, such as spaCy, have models that recognize entities and what they represent in the real world or based on a fictional context. E.g., “Naira” in some contexts might mean a person’s name while it could just be referring to a unit of currency in some other context. NLP frameworks try to predict if the entity is a person or a currency, or something else given the context of the sentence. However, the models available in most of the frameworks are general-purpose pre-trained models and do not have the semantic knowledge that will be sufficient for this project. The other option would be to train a model on specific types of stories to recognize entities specific to those types of stories. As Kamath and Wagh (2007) explained in their work, the challenges faced with current approaches to NER is that their performances are only optimal when used on the specific types of data they’ve been trained on.

Consequently, in this work, the system takes in entities and their names (labels) directly as the second input. It also makes sense to take labeled game character as input because the process doesn’t require much extra work from the game designer, as game world characters will also be readily available either through another procedural generation process or manually before the development stage. Even on online game databases, the

entities (locations, characters, weapons, etc.) are readily available. There are no restrictions on entity types and entity labels. Both are determined by the user of the system. The example below shows how entity labeling should be inputted into the AQPG system.

Entities = {

Slayer: Player,

Hell: Location,

Khan Maykr: NPC,

Sentinel Prime: Location,

BFG 9000: Weapon

Zombies: Enemy,

Archvile: Enemy,

}

Only one player can be labeled as the *player* or else the system will take the first player found in the list as the main character.

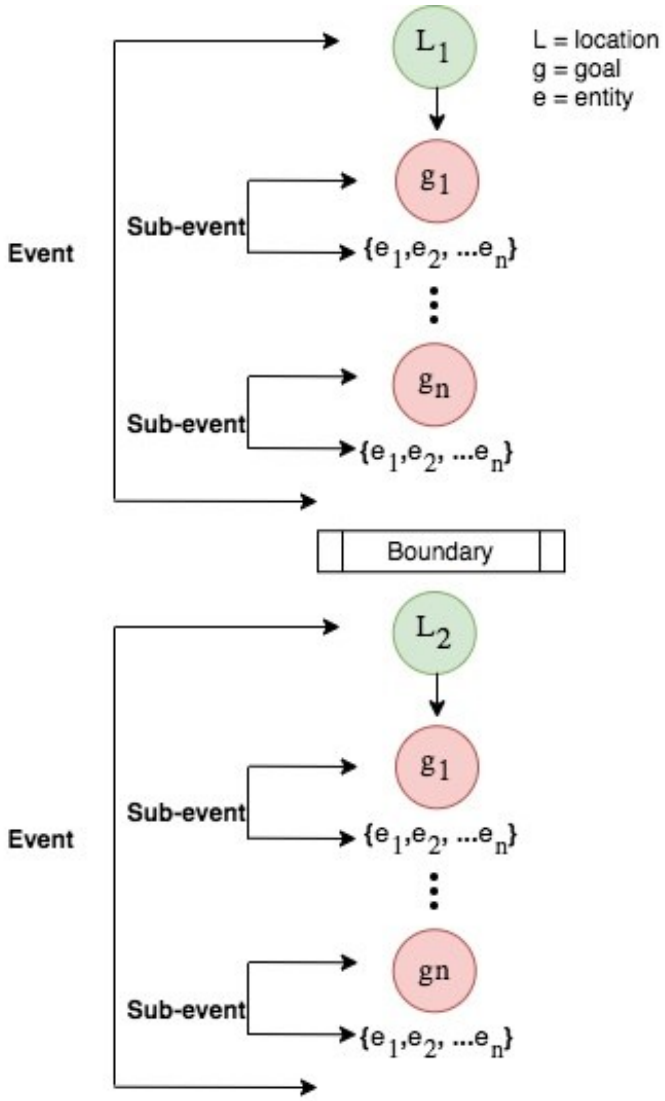

Figure 2.The Event Model

3. 3 The Event Model.

It is worth discussing the event model since both the events and sub-events use computation of the model for their respective functions.

As discussed earlier, identifying event boundaries is the first and most important aspect of ES because it denotes the end of a current event and signifies the beginning of a new one. However, without constructing an event model, this will not be possible.

As discussed in chapter 2, several previous projects have focused on learning event models in humans, but the computation of an event model has posed a big challenge over the years. Baguley and Payne (2000) suggest that event boundary activation uses a hierarchy of precedence in activating boundaries. Temporal Inference is discussed broadly in some studies like Anderson (2005) while Khemlani, Harrison, and Trafton (2015) suggest that location takes precedence over other cues. More recent works on segmentation have come up with models that are more temporally sensitive (Modi, 2016; Reynolds, Zacks, & Braver, 2007).

In contrast, I decided not to consider temporal changes given how most game plots are narrated. In most role-playing game plots, new journeys seem to occur in new locations (L1, L2, … L*n*), often without explicit mentions of time. E.g. "*In Rome, Thorton meets Madison Saint James, with whose help he discovers…"* ("Alpha protocol", n.d) . The start of a new journey (E.g., getting to the location Rome above) often comprises a related chain of actions. The actions of the player entity at a new location depict the sub-goals (g1, g2… g*n*) that a player must accomplish to achieve an overarching goal . Each subgoal (g) is detected by the AQPG when a the player entity performs a new action in the story. The subgoals can have a set of entities {e1, e2,…e*n*} that are available while each subgoal is being accomplished. Examples of these entities are characters, items, tool, animals etc. Each location can have different subgoals and each subgoal can have different entities depending on how the story is written.

For the reasons explained above, I propose in this thesis that the best order of precedence for marking event boundaries in quest plots would be Goal(G)-> Location (L)-> goal (g)-> elements (e) available during the goal. The ‘Goal’ is the final goal of the quest game (e.g., in the Aladdin world the 'Goal' might be getting the lamp and there will be many sub-goals along the line before getting to the last quest of getting the lamp). Since the overall Goal will not change in the game world, there’s no need to include it in the model.

This model does not focus on modelling events representation in a human-like way. Rather, the event model of this thesis was designed in a way to ensure that at the end of a sub event segmentation, the system has all the elements it needs to generate a complete domain file and problem file for a planning agent.

3.4 Segmentation based on location change

I wrote the segmentation algorithm to follow the event model that I designed for this project. Here at the event segmentation stage, the story data is broken down into a set of events. Identifying the event boundary is the most important function of the segmentation algorithm.

To achieve this, firstly, the segmentation algorithm creates a list all the entities that are labeled ‘location.’ See Fig. 4 for an example.

```
['space station', 'kul elna', 'mortuary temple', 'domino city', 'kame game', 'domino way', 'high school',
'mall', "devlin's cafe", 'plaza', 'kaiba land', 'kame game', 'stadium', 'plana realm', 'earthport', 'palac
e']
```

Figure 3. Example of locations retrieved from yu-gi-oh in a list format

The segmentation algorithm then runs through the story data in search of every point where those locations are mentioned. To ensure to some extent that the location depicts that the main character (player) of the story is at a new location, the algorithm ensures that:

a. At least one subgoal (g) is identified in the texts before a new L comes up.
b. A new location (L) has one preposition as a child in its dependency parse tree. E.g., ***At** the **space station**, the computer has almost finished reassembling the Millennium Puzzle*
c. An action marks the presence of a subgoal (g) is performed by the main character.
d. The boundary is triggered only when a new L meets the above-mentioned requirements.

The search ends by returning the point at which every location appears in the story and in the order in which they appear. After the search ends, the algorithm can now perform a segmentation of the whole story input.

A segment is constructed by the algorithm in the following way.

a. The first event starts at the beginning of the story (point zero) and ends at the end of the last full sentence before the next boundary.
b. The last segment starts at the last relevant location and ends at the end of the story.
c. The segments in between the last and the first, start at the end of the sentence in the last segment to the next location.
d. All events contain full sentences that end with full stops as you'll see in examples at the end of this chapter.

The output of this stage is a series of story segments that I call ‘events’. These are the kind of events that Barker and Wright (1954) referred to as ‘long events’. I show some examples of the output of the events in later parts of this section.

3.5 Segmentation based on goal change

The segmentation based on goal module is the most important part of the system. Its inputs are the segmented events explained above, and its outputs contain almost everything needed by AQPG to generate a domain file.

At this second segmentation module, a sub-segmentation algorithm, also written by me, takes the events from the first segmentation module and breaks then down into series of sub-events. As one of its many functions, this module implemented SpaCy NLP library for part-of-speech-tagging and dependency parsing. SpaCy is an open-source NLP library published under the MIT License. Its features include part-of-speech tagging, dependency parsing and text categorization. However, the only features of SpaCy needed in this work is its part-of-speech tagging and dependency parsing.

In the event model, a new sub-event is identified inside an event when a new goal has been detected by the sub-segmentation algorithm. This new goal is represented as starting when the Player entity makes a new action in an event. This means that there is one subgoal per action taken by the player. The example below shows how the sub-segmentation algorithm works.

Algorithm:

a. For Each segment, the algorithm searches for actions performed by the player. This is done by using part of speech and syntactic dependency tags provided by SpaCy.
b. If the noun subject (nsubj) of an action (verb) is the player character, then the action is relevant. This is the only thing SpaCy is needed for in the module.
c. For each segment, a subsegment starts when the player performs an action and ends at the end of the sentence containing the action. I call these 'sub-events'.

These resemble the events that Barker and Wright, (1954) referred to as 'brief'.

For example, in the sub-event below;

"*Slayer travels to Hell*."

The program detects that the Slayer travels in this statement at the end and then takes the sentence from where it begins to the end and stores it as one sub-event.

d. Detect player's actions in the event.
e. Store each action's respective segment numbers and location point in a dictionary format. This is what it looks like:

{1: [['returns', 2]], 3: [['travels', 13], ['kills', 73]], 4: [['destroys', 6], ['makes', 47]], 6: [['travels', 28], ['finds', 127], ['kills', 161]], 7: [['uses', 36]], 9: [['kills', 145], ['walks', 173]]}

Example Data: Doom's day's Plot

Eight months after the events of 2016's DOOM, Earth has been overrun by demonic forces, wiping out 60% of the planet's population, under the now-corrupted Union

Aerospace Corporation. What remains of humanity has either fled Earth or have banded together as part of ARC, a resistance movement formed to stop the invasion but have gone into hiding after suffering heavy losses. The Doom Slayer, having previously been betrayed and teleported away by Dr. Samuel Hayden, returns with a satellite fortress controlled by the AI VEGA to quell the demonic invasion by killing the Hell Priests; Deags Nilox, Ranak, and Grav. The priests serve an angelic being known as the Khan Maykr who seeks to sacrifice mankind. The Slayer teleports to a destroyed Los Angeles and kills Deag Nilox, but the Khan Maykr teleports the two remaining priests to unknown locations, forcing the Slayer to continue searching. After retrieving a celestial locator from the Sentinel world of Exultia, the Slayer travels to Hell to retrieve a power source from a fallen Sentinel known as the Betrayer. He warns the Slayer that humanity's time has come before giving him the power source and a special dagger. VEGA directs the Slayer to a citadel in the Arctic where Deag Ranak has taken refuge, and the Slayer kills him after defeating his guardian Doom Hunters. In response, the Khan Maykr moves Deag Grav to a hidden location and accelerates the invasion of Earth. Forced to change tactics, the Slayer destroys the Super Gore Nest in central Europe, where the invasion began. With no leads on finding the last Hell Priest, VEGA suggests finding Dr. Hayden, who knows the location of Deag Grav. The Slayer makes his way to an ARC compound where he retrieves Hayden's robotic shell, as well as the Crucible. (Fandom, n.d.)

Example Output: Event Segmentation

Event 1: Eight months after the events of 2016's DOOM, Earth has been overrun by demonic forces, wiping out 60% of the planet's population, under the now-corrupted Union Aerospace Corporation. What remains of humanity has either fled Earth or have

banded together as part of ARC, a resistance movement formed to stop the invasion but have gone into hiding after suffering heavy losses. The Doom Slayer, having previously been betrayed and teleported away by Dr. Samuel Hayden, returns with a satellite fortress controlled by the AI VEGA to quell the demonic invasion by killing the Hell Priests; Deags Nilox, Ranak, and Grav. The priests serve an angelic being known as the Khan Maykr who seeks to sacrifice mankind.

>>> BOUNDARY (New location Los Angeles)

Event 2: *The Slayer teleports to a destroyed Los Angeles and kills Deag Nilox, but the Khan Maykr teleports the two remaining priests to unknown locations, forcing the Slayer to continue searching.*

>>> BOUNDARY (New Location 'Hell')

Event 3: *After retrieving a celestial locator from the Sentinel world of Exultia, the Slayer travels to Hell to retrieve a power source from a fallen Sentinel known as the Betrayer. He warns the Slayer that humanity's time has come before giving him the power source and a special dagger.*

>>> BOUNDARY (New location 'Super Gore Nest')

Event 4: *Forced to change tactics, the Slayer destroys the Super Gore Nest in central Europe, where the invasion began. With no leads on finding the last Priest, VEGA suggests finding Dr. Hayden, who knows the location of Deag Grav.*

>>>> BOUNDARY (New location 'ARC Compound')

Event 5: *The Slayer makes his way to an ARC compound where he retrieves Hayden's robotic shell, as well as the Crucible.*

Example Output : Sub-Event Segmentation

Event 1:

1. *The Doom Slayer, having previously been betrayed and teleported away by Dr. Samuel Hayden, returns with a satellite fortress controlled by the AI VEGA to*

2. *quell the demonic invasion by killing the Hell Priests; Deags Nilox, Ranak, and Grav.*

Event 2:

1. *The Slayer teleports to a destroyed Los Angeles*

2. *The Slayer kills Deag Nilox, but the Khan Maykr teleports the two remaining priests to unknown locations, forcing the Slayer to continue searching.*

Event 3:

1. *After retrieving a celestial locator from the Sentinel world of Exultia, the Slayer travels to Hell to retrieve a power source from a fallen Sentinel known as the Betrayer.*

2. *Slayer retrieve a power source from a fallen Sentinel known as the Betrayer*

Event 4:

1. *Forced to change tactics, the Slayer destroys the Super Gore Nest in central Europe, where the invasion began.*

Event 5:

1. The Slayer makes his way to an ARC compound where he retrieves Hayden's robotic shell, as well as the Crucible

3.6 Generating a Domain File

Planning tasks specified in PDDL are separated into two files: the domain file and the problem file. Planning Domain Definition Language (PDDL) is a standard encoding language used for planning tasks. The domain files contain all possible actions in the world, the conditions that must exist in the world for each action to be available for execution and the effects of the action on the state of the world. The problem file contains the objects, a description of an initial state of an environment and a representation of goals to be achieved.

The idea in this project is that segmented events from natural language text can generate these actions, preconditions and effects for a domain file just as humans use event segmentation while perceiving an event to reason about the actions and consequences (effects) of those actions. The system infers preconditions and parameters from the objects and other characters (humans, animals etc.) that are present when an action takes place.

All the contents in the domain file are automatically generated by algorithms written for this project. To generate those contents, the algorithm needs to first extract relevant information from the sub-events and then decide on how to represent that information in PDDL format.

A. Extracting Contents

To generate the contents needed for a domain file, the system extracts information from the outputs of Sub-Event Segmentation, the sub-events and a dictionary containing actions and the segment in which they belong.

For each sub-event, I designed an extraction algorithm that searches for all entities in sub-events and returns corresponding labels of entities and the action in the sub-event. This makes sure to an extent that the entities extracted are important to the action.

1. Sub-event 1: *Slayer teleports to a destroyed Los Angeles*

   Finds : PLAYER slayer + ACTION teleports + LOCATION Los-Angeles

   Stores: Entities[Player, Location] Action[teleport]

2. Sub-event 2: *Slayer kills Deag Nilox with a gun, but the Khan Maykr teleports the two remaining priests to unknown locations*.

   Finds: PLAYER slayer + ACTION Kills + NPC Khan Maykr + PRIEST Deag Nilox + WEAPON Gun

   Stores: Entities[Player, NPC, Weapon, Priest] Action [Kill]

It is worth noting that all actions are lemmatized before they are stored back at this stage. Lemmatization is the process of removing endings of words and returning only the base words. E.g., Result of the lemmatized word *acted* is *act*. Likewise, *colors* will be *color after lemmatization.*

At this stage, after storing all the extracted contents of sub-events in a list, e.g. *[[Player, Location], teleport],[[Player, Weapon, Priest], Kill]*. A function *unduplicate(lst)* takes

the list and converts it to a python programming language's dictionary data type. This conversion removes all duplicates, leaving only one occurrence of each item in the list because python's dictionary data type doesn't support duplicate keys. After shedding the duplicates, the function converts the dictionary back to a list data type and returns it. This process of removing duplicates is so that the system doesn't construct redundant planning operators.

The job at this stage is almost done, but the extracted entities are not completely useful without converting them to some representation that the domain file needs. Therefore, another function assigns an abbreviation to each entity.

To assign an abbreviation to the entities, the function uses the first three letters of the entity label. This is to reduce the chances of having different entities being assigned the same label. If the abbreviation has already been assigned, the function uses the first four letters. For example:

All Entities from the two examples of sub-events discussed above:

1. Entities: [Player, location,]
2. Entities: [Player, NPC, Priest]

Entities and abbreviations:

1. [Player ?pla, Location ?loc]
2. [Player ?pla, NPC ?npc Priest ?pri]

B. Generating Predicates

The domain file contains predicates and actions. Each predicate represents a property of a state. E.g. (player ?pla) is one of the properties a state can have, likewise (teleport ?pla ?loc). A valid predicate must be a state that can exist in the world. The list of predicates describes which objects can have a particular predicate in a certain state. E.g. (teleport (?pla ?loc)) in the list of predicates means that *teleport* should be available with a combination of ?pla and ?loc. The first set of predicates define predicates available to all elements that exist in the world. It is a representation of all entity labels and abbreviations that are found in all the sub-events gotten in the above section.

```
(:predicates
   (player ?pla)
   (facility ?fac)
   (information ?inf)
   (character ?cha)
   (building ?bui)
   (enemy ?ene)
   (item ?ite)
   (weapon ?wea)
   (animal ?ani)

)
```

This is followed by a list of the effect of actions and entities connected to the action. E.g. *Tom will kill Sarah* will have an effect that player entity Tom killed an animal, Sarah, if Sarah is labelled as an animal. Meaning that "(killed ?ani)" is a state that can exist in the world and will be added to the list of predicates. The process of determining effects is described below under 'Generating actions'.

```
(:predicates
        (player ?pla)
        (facility ?fac)
        (information ?inf)
        (character ?cha)
```

(building ?bui)
(enemy ?ene)
(item ?ite)
(weapon ?wea)
(animal ?ani)

(killed ?ani)

(pushed ?cha)

)

C. Generating actions

Actions are also called planning operators. They are the possible actions that the planner will tell the player character to perform. Action representations are composed of the action name, parameters, preconditions and effect.

```
(:action ACTION_1_NAME
  [:parameters (?P1 ?P2 ... ?PN)]
  [:precondition PRECOND_FORMULA]
  [:effect EFFECT_FORMULA]
 )
```

Figure 4. Template of a planning operator

Action name: All action names are the actions taken by the player character in each sub-event. For each sub-event, a player performs an action and the action name is extracted and lemmatized.

To lemmatize an action, I downloaded an open-sourced text file from Node Box's GitHub page. NodeBox is a Mac OS X application that allows designers to create 2D visuals and export them as a PDF or a QuickTime movie. Node Box has a linguistics module on its

GitHub page that contains verbs in every tense they can appear in English natural language text. Then I converted the file to a comma-separated value (csv) file.
Following that, I defined a function that takes an action word as a parameter and searches through the file to retrieve the action word in its present tense.

Parameters: Parameters are usually placeholders for what to expect in the preconditions. For example, a parameter *?pla* could mean that the preconditions should expect to have player as one of its elements. The equivalent precondition element would be represented as (*player ?pla).* Parameter elements are free form in the sense that a parameter element *?npc* could be a placeholder for a precondition element *(character ?npc).* In this work though, I maintained uniformity, that is, *?char* would only represent (character *?char).* The parameters are encoded directly from the entities they represent.

Algorithm (param_encode) explanation:
Recall the stage where entities are extracted from events, where a function searches for all returned entities from the sub-events and returns corresponding labels of entities that match. Those labels are used by the system to construct parameters for actions here. For example, consider one of the outputs from the section above where I discussed how contents are extracted from sub-events. For example, if this was one of the outputs:

Entities[Player ?p, Location ?loc Location ?loc], Action[teleport].

This means that the parameters for the action *teleport* are player and location1 and Location2. The param_encode algorithm will encode the parameters as :*parameters (?p ?l1 ?l2).* If separate entities of the same label appear twice, the system will add a counter. E.g, Event : "Tom crossed the forest into the plains" will have :*parameters(?pla), (?loc1), (?loc2)*

Preconditions: These are the set of predicates which must be true in order for the action to be applicable. The parameters available determine how the precondition algorithm will construct the preconditions. I already explained that parameters are like placeholders for elements in the precondition. Therefore, given the same output from the extraction process:

[Player ?pla, Location ?loc Location ?loc], Action[teleport].

The algorithm just takes *[Player ?p, Location ?loc Location ?loc] and converts it to (and (?player ?p) (location ?loc1) (location ?loc2))*.

Note: In a previous example, this sub-event:

Slayer kills Deag Nilox with a gun, but the Khan Maykr teleports the two remaining priests to unknown locations outputs : *Entities[Player, Priest, Weapon, NPC] Action [Kills]*.

Here, the system will incorrectly require any NPC as one of the preconditions of the action 'kill', because an NPC *Khan Maykr* was present in the same sub-segment, even though Khan Maykr is not necessary for the action to take place. This is one of the current drawbacks in the system although it will not affect the plan because the effect of the action is only going to be on *Deag Nilox*.

Effects: Effect is the consequence of an action, which takes the world state into a new state. Determining causal effects through NLP has never been easy. Most of the attempts work to identify causal effects. For example, Girju and Moldovan (2002) have used

manually constructed linguistic and syntactic rules on small and domain-specific data sets. Effects can be implicit and explicit. The plan for generating effect is as follows.

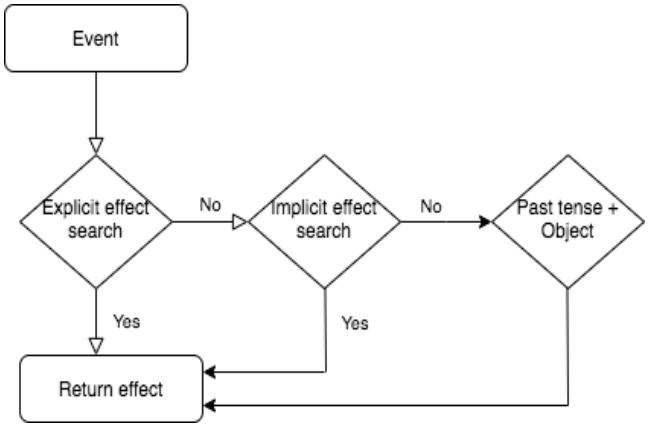

Figure 6. Flow diagrams of effect generation

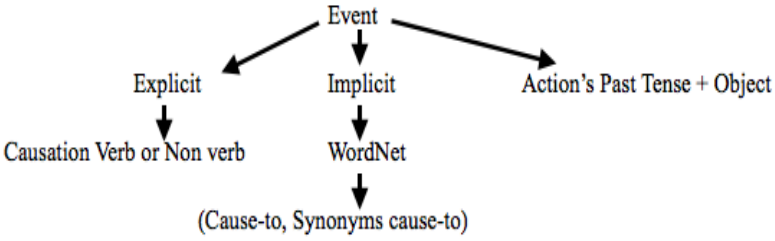

Figure 5. Second Flow diagrams of effect generation

a. Explicit effects

A customized search algorithm determines if the event states the effect of the action explicitly. If the effect is stated explicitly in a sentence, it often contains two components, the cause and the effect. E.g., "I stepped on the cockroach. As a result, it died." Explicit effects in many cases include key words like cause, consequence, etc. By writing a search function for causation verbs, causation and other non-verbal patterns like because, due to and when. The system might be able to identify if an explicit effect is stated. Here a list of causation verbs from Girju and Moldovan (2002) which contains 60 ambiguous and not so ambiguous causation verbs.

Low ambiguity High frequency	Low ambiguity Low frequency	High ambiguity Low frequency	High ambiguity High frequency
induce	stir up	create	start
give rise (to)	entail	launch	make
produce	contribute to	develop	begin
generate	set up	bring	rise
effect	trigger off		
bring about	commence		
provoke	set off		
arouse	set in motion		
elicit	bring on		
lead (to)	conduce to		
trigger	educe		
derive (from)	originate in		
associate (with)	lead off		
relate (to)	spark		
link (to)	spark off		
stem (from)	evoke		
originate	link up		
bring forth	implicate (in)		
lead up	activate		
trigger off	actuate		
bring on	kindle		
result (from)	fire up		
	stimulate		
	call forth		
	unleash		
	effectuate		
	kick up		
	give birth (to)		
	call down		
	put forward		

Figure 7. Girju and Moldovan's Ranking of Caussation Verbs

For example; "I killed boars in the jungle, causing them to die". The search algorithm will identify the causation verb *cause* from the list in Fig 5, boar which is an entitiy *anim* and cause effect *die* will be extracted for an action killed.

:effect (died anim?)

b. Implicit Effects

Implicit effects are complex to analyze. It means that the effect of cause needs to be generated from inference based on semantic analysis. If an effect is not found explicitly, another customized search algorithm takes the action but this time runs it through WordNet on python's Natural language processing Tool Kit (NLTK), in an attempt to get the implicit effect of the action. WordNet explicitly uses CAUSE-TO as one of its semantic relations. When a definition of a verb is called on WordNet, word net returns a list of sentences that defines the verb, explains the verb and also tries to explain the verb's semantic relations.

The implicit search algorithm will infer an effect of the verb in question by extracting the verb after the sentence with *cause-to* from word net's list of definitions.

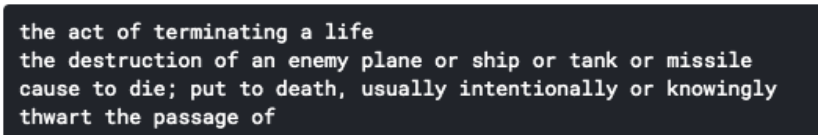

Figure 8. WordNet Cause-to Semantic relation

From the figure above WordNet returned a list of definition for the verb kill, the algorithm will identify that the *cause-to* is located in the third sentence and search for the causation in it, in this case, *death.*

As said earlier, implicit causation is very complex and most verbs on WordNet do not even have CAUSE-TO in their semantic relations, if the CAUSE-TO is not found, the algorithm then takes the action, gets the synonym with the closest similarity to the action and performs the same CAUSE-TO search operation.

c. Past tense and objects

Lastly, the algorithm converts the lemmatization of the action to past tense and connects it with the object the player character acted upon.

E.g., Tom will Kill a lion. The action Kill turns into Killed and linked to the entity (ani) it occurred to. E.g., effect: (Killed ?pla ?ani) will be appended to the list of effects of that action. Getting the object of the action uses a similar strategy as getting the actions performed by the player entity. The function to get effect uses the same dependency tags that determined the player's actions to get the objects of the same action. The function picks out the objects of the action and arranges them on a first-in basis. The first element is the most important object while the last element to be stored is the last object. In most cases, there is only one object.

The Past tense and objects strategy is used regardless of the outcomes of the first and second effect generation strategies because it has a guaranteed outcome. The cost of each planning operator is generated randomly between 1 to 5. At the end of this, the system will have enough information to generate a domain file.

A cut out from the domain file generated by AQPG system from the story plot of yu-gi-oh game showing what the result looks like:

```
(define (domain yugiohland)
    (:requirements :action-costs)
    (:predicates
        (player ?pla)
        (location ?loc)
        (game ?gam)
        (box ?box)
        (character ?cha)
        (information ?inf)
        (item ?ite)
        (saved ?npc)
        (battle ?bat)
        (antagonist ?ant)
        (began ?bat)
        (added ?gam))

    (:functions
        (total-cost))

   (:action save
   :parameters (?pla ?ite ?cha )
   :precondition (and (player ?pla) (item ?ite) (character ?cha) )
   :effect (and (saved ?cha) (increase (total-cost) 1))
   )

   (:action add
   :parameters (?pla ?loc ?box ?gam ?ite )
   :precondition (and (player ?pla) (location ?loc) (box ?box) (game ?gam) (item ?ite) )
   :effect (and(added ?gam)(increase (total-cost) 2))
   )

   (:action begin
  :parameters (?pla ?ant ?bat )
  :precondition (and (player ?pla) (antagonist ?ant) (battle ?bat) )
  :effect (and(began ?bat)(increase (total-cost) 2))
  ))
```

Figure 9. A cut out from Yu-Gi-Oh's generated domain file

Note: The original domain file contains 24 planning operators

3.7 Generating a Problem File

The problem file describes the initial state of the world and the goal criteria for specific scenarios in the world. The initial state describes all that exists in the story world, such as all characters, locations, items, etc., while the goals describe what the planner needs to create plans to achieve.

The template of a problem file looks like this.

```
(define (problem <problem name>)
   (:domain <domain name>)
   <PDDL code for objects>
   <PDDL code for initial state>
   <PDDL code for goal specification>
)
```

Figure 10. Problem File Skeleton

a. The <problem name> identifies the planning tasks, and the <domain name> must match the domain name in the corresponding domain file.

b. The objects are encoded as all the entities that exist in the story world.

```
(:objects you nest ballista bfg-chaingun vega chainsaw shotgun doomblade gun canon hook
rifle rocket-launcher crucible unmaykr pistol hell exultia cultist-base super-gore-nest)
```

Figure 11. Objects Representation

This is relatively easier to generate as the system would already have a list of the entities before this stage. The only adjustment is converting the main character entity to be represented as ‘you’.

c. Initial State: The system represents the initial state as the entities and their corresponding labels. In complex planning problems, the problem file often contains a representation of the relationship between the entities in the world too. However, in this thesis, the system does not create a complex definition of the world, it only creates a representation of what each entity in the world represents.

```
(:init
(player you)
(weapon shotgun)
(weapon bfg-chaingun)
(location hell)
(location mars)
(location los-angeles)
(facility cultist-base)
(facility satellite-fortress)
(enemy buff-totem)
(enemy tentacle)
```

Figure 12. Initial State Representation.

d. Goal Specification: The goals are statements that must come true in the world. In this system, the goals are sets of predicates. The goals are different states that the agent wants to be true in the world, therefore, since predicates are different states in the world, the system can choose its goal from the sets of predicates.

Simulation: The goals are simulated by a random goal generator. Every time the system runs, a new goal is generated by a goal simulator algorithm. The algorithm takes in objects and state predicates as inputs and outputs random goals of random length. Below is a rough outline of the algorithm.

- Set O, the current state of the problem. I.e., the set of literals in the initial state of the problem file, e.g., ‘location mars’
- Set P, state predicates, e.g., ‘Killed ?pla ?cha’ and not entity predicates such as ‘player ?pla’

1. selects random number (of goals) x between 1 and 10 .
   x = *2*
2. selects x number of state predicates at random
   ‘Killed ?pla ?ene’, ‘Destroyed ?pla ?loc
3. replace predicate labels with corresponding initial state objects
   ‘Killed you gladiator’ , ‘Destroyed you los-angeles’
4. remove player object
   ‘Killed gladiator’ , ‘Destroyed los-angeles’
5. return goals .
   (:goal (and (killed gladiator) (destroyed los-angeles))

3.8 Planning Algorithm

After generating the PDDL files (problem file and domain file), the system can now proceed to implement a planning algorithm. The system is not built to generate complex plans but rather plans that can be translated into quests. The PDDL files are also expected to work regardless of how the planning is implemented. I.e., whether the planning is done using single or multiple agents.

FastDownward: The system uses FastDownward planner as distributed under the GNU General Public License. The FastDownward planner is a classical planning system based on a heuristic search (Helmert, 2006). It deals with planning problems encoded in PDDL2.2, a standardized format for planning which allows specifying the domain files

and problem files. The PDDL allows us to define the AI problem while the planner reads the PDDL and uses it to decompose and solve the problem.

FastDownward computes its heuristic functions through hierarchical decompositions of planning tasks (Helmert, 2006). The reason why it's been used is that it supports many heuristic functions such as the A* search algorithm that is used in this project. It also supports Action Description Language (ADL) conditions and effect which allows numerical values to be used in the domain file which is instrumental in creating a weight of actions of agents and also manipulating the weight of actions. It also has added advantages such as support for numerical values uses alternative representation called multi-valued planning tasks, which makes many of the other implicit constraints of a propositional planning task explicit.

Implementation:

For the implementation, I ran the planner with an A* algorithm as it supports guiding the plan search with heuristics and the cost metric. The heuristic aims to minimize the cost of reaching the goals. At every instance of the AQPG system run, FastDownward takes in the auto encoded PDDL files, decomposes and returns a plan to achieve the goal specified in the problem file. Since the goal changes every time the system runs to generate files, the system generates different plans every time.

Chapter 4

Results and Evaluation

In the previous chapter, I discussed how the system works to achieve the goal of generating planning files for an AI planner. I also explained how the planner generates the files to create quests in form of plans. In this section, I discuss the results and how the system's performance is evaluated.

4.1 Overview

I used five different story plots to test the system. 3 of the stories were retrieved from fandom.com while 2 were retrieved from Wikipedia. One of the two stories from Wikipedia is not a game plot, it is the storyline of the movie Aladdin (2019).

	Doom Eternal	**Alpha protocol**	**Yu-Gi-Oh! The Dark Side of Dimensions**	**Aladdin**	**Doom Day**
paragraphs	7	4	10	7	7
words per document	818	844	1519	687	591
source	fandom	Wikipedia	fandom	Wikipedia	fandom

Table 1. Statistics on various stories used as data for the system

The first part of this section evaluates the Event model by testing the segmentation processes, the second part evaluates the planning files and quests generated by the system.

4.2 Event Model

The purpose of the event model in this work is to segment stories into discrete events. Segmentation works are not easy to compare directly given that some segmentation methods may be based on topics, while others can be based on semantic structures. Most recent works on text segmentation focus on the use of machine learning techniques and a large corpus of data to divide documents into contiguous segments. Those works (Chen et al., 2009; Hearst, 1994; Utiyama & Isahara, 2001; Brants et al., 2002) use very specific types of datasets that are peculiar to their scope and domain to evaluate their methods. For example, Chen et al. (2009) proposed a method that requires all the documents in their dataset to be of the same topic.

I feel that it will serve no purpose to compare the performance of this work's segmentation model to these systems, given that the text segmentation for the use case of this system is very different. The text segmentation here is based on the ordering of events, while previous works on text segmentation focus on topics, words, intents, etc. Therefore, there is no basis for comparison as the segmentation here is meant to serve an entirely different purpose. The segmentation here is based on change of location and change of goal modeled around one character in a story.

Because the event model was designed for the computation at both segmentation and sub-segmentation stages, if the system was successful in implementing the model at both stages, then it will prove that the model was instrumental in auto encoding the planning files. In order to determine the accuracy of the computation, I came up with an error rating method to determine the accuracy of the segmentation process. The method uses stories that I have manually segmented based on the event model, to compare with the segments created by the system.

4.2.1 Error rating

Error rating involves comparing the accuracy rates of human segmented data to automatic software. We are always trying to reduce the error rate in NLP works where error rates need to be evaluated (Riedl & Biemann, 2012), and even in segmentation of images works like Rankin et al. (2010). A low error rate would mean a high accuracy, that is, minimizing false positives.

There is no standard metric for error rating. Many studies decide on what their definition of accuracy is. As Brill (1992) explained, accuracy metrics for segmentation are subjective and dependent on the domain. Also, because I have found no studies where segmentation was computed based on series of events in a natural language text, I could not compare the segmentation modules in this work directly with other works.

To evaluate the segmentation modules, I manually segmented 5 stories using the event model and came up with two error rating formulas. The first formula is for Exact Boundary matching (*M*) while the other is for segment counts (*S*). *M* tries to evaluate if the starting points and endpoints of each segment are accurate. That is, if the starting point and endpoint is the same as the human labelled starting point and end point. *S* on the other hand evaluates the accuracy of the number of segments the series of events were split into.

$$M = \sum_{i=1}^{n} e_i / n$$

$$\text{Where: } e = \frac{\text{(correct starting points + endpoints)}}{\text{(total starting points + endpoints)}} * 100$$

and n = number of stories

S = percentage of difference between total number of segment and the system's segment count.

I computed *M* and *S* for both segmentation and sub-segmentation modules.

4.2.2 Segmentation Module

To analyze how the system makes ratings for each story, I'll explain an example of the first story used. The system's segmentation module was first tested on Doom Eternal's

(DE) Plot. DE is an online role-playing game with single and multiplayer options. In DE's storyline, the main character is called 'slayer' and the segmentation here is executed around the slayer character as the player. The story and its labeled entities were entered as inputs into the system. The segmentation module recognized 10 segments. Each segment is a complete meaningful sentence ending with full stops. Of all the 10 segments, 9 of them correctly segments the player character being at a new location. The one segment that the system failed to account for counts toward the error rates, as it will reduce the segment count to 9 instead of 10. This makes the *S* score 90%.

Boundary matching means that the human-marked starting point of the event is the same as the system's, and also the human marked endpoint of the event should be the same as the system's marked endpoint. Therefore, for every available segment, if the marked beginnings correlate, it counts as one point, if the marked endings correlate, it counts as another point and likewise, if any of the points are marked wrongly, it counts as an error. At the end of the comparison counts, the percentage of the numbers the system got correctly is its boundary matching accuracy.

This method was inspired by the work of Koshorek et al. (2018), who developed a segmentation model that segments text as a supervised learning task. They evaluated the performance of their method by using human segmentation as the upper boundary baseline and used random segmentation as the lower baseline. The table below summarizes the accuracy of the segmentation module.

	M (%) Total = 120	*S*(%) Total = 60
Segmentation Module	98.3% = 118	95% = 57
SpaCy	68.3% = 82	47.5% = 28.5

Table 2. Segmentation accuracy rates

Here, I used SpaCy's random segmentation as my lower boundary baseline and the human segmentation (event model) as my upper baseline. SpaCy had a boundary matching score of 68.3%. Approximately 45% of that came from SpaCy matching the beginning of events but it did poorly on matching the endpoint of segments. SpaCy's segmentation count score was worse. The system's segmentation's matching score of 98.3% and segment count score of 95%, shows that it produced a marginal error when compared to the human segmentation based on the proposed model. This result means that the system is suitable to carry out its intended purpose of executing the first segmentation phase.

4.2.3 Sub-segmentation Module

I also wanted to test if the system correctly implements the model at the sub-segmentation stage, so I did manual sub-segmentation on the same 5 datasets. The results are in the table below.

	M(%) Total = 170	*S*(%) Total = 85
Sub-segmentation module	92.3% = 157	94.1% = 80
SpaCy	30.0% = 51	35.8% = 30.5

Table 3. Sub-segmentation accuracy rates

For the system's sub-segmentation, the results were similar to the segmentation phase, although the scores were a little bit lower. SpaCy also performed worse in this phase on both scores. The system's score again proved to be positive, having an Exact Boundary Matching and Segment count score of 92.3% and 94.1%, respectively.

4.3 Planning

Being aware that this method of generating plans is novel and autogenerating plans for this domain is also unusual, I analyzed whether the plans generated by the system are executable and useful for quest generation.

Nabil and David's (1989) research on evaluating AI planners describes what a correct plan is. They explained that a plan produced by a planner is correct if the execution of the plan, starting from the initial state of the world transforms the world to a goal state.

> *"We start with a noncontradictory initial state of the world and we are supposed to make our way to a non-contradictory final state of the world."*

Therefore, if we are to implement a planner with a set of planner files, if the planner finds a solution to the problem, both the planner files and the planner are sound. Also If an

intermediate world state shows any inconsistency, the plan is said to be bad. If the plan also does not reach the final desired world state, the plan is categorized to be bad.

4.3.1 Quality and executability of planning files

The system was used to automatically generate 5 domain files and 5 problem files from 5 data inputs. After the successful generation of files, each couple of files were used as input to the FastDownward Planner. Then the planner created a solution to attain the goals encoded in the problem files.

If the planning files were encoded wrongly, the system would be expected to break. If the process completes without breaking, then the files were created correctly. At the end of the execution, all files that were used as input executed completely without any break in the system. This proved that the files generated by the system are completely executable as planning files.

Furthermore, usage of planning files differs from domain to domain. After implementing a planning algorithm, if the execution process does not break until the end of the process, it means that both planning files were created correctly. To assert that the plans were quality enough to generate quests, the planner must find a solution within the planning operators and states to reach the goals encoded in the problem file. If the planner returns a file with the name *sas_plan,* it means that the automatic generation of the planning files was a success and a solution was found for the problem. For the 50 iterations of planning

for each story's planning files, the planner returned a *sas_plan* file every time (250). This means that the planning file generation had a success rate of 100%.

4.4 Quests

As explained in previous chapters, quests here are defined as series of steps an agent must take to achieve its goals. In multi-agent planning systems, the sequence and coherence of quests are determined by a collaborative effort of multiple agents. In single-agent planning systems, the agent determines the sequence and coherence of quests. Because the work of this proposed system focuses on creating the planning files for those systems, I am more concerned about evaluating whether the plans that the system generates are (1) useful to generate quests and (2) are similar to human-authored quests.

In the planning section above, I have described that the automatically generated files were successful in generating quests. The quests are plans that contain series of actions that take the initial state of the world to its goal state. Here, I am trying to verify that the quests are the types of quests authored by humans.

Doran and Parberry (2010) worked on human-authored quests of commercial games. They made a list of actions and described them as complete. In the sense that all of the human quests they studied falls under a combination of those seven categories of actions.

Section	Action	Meaning
1	**[Attack [xN] {NPC, Item}]**	Damage an entity
2	**[Talk NPC]**	Talk to an NPC
3	**[Assemble Item]**	Assemble a new item from parts
4	**[Give Item to Entity]** **[Take Item [from Entity]]** **[Trade Item for Item with Entity]**	Trade items with an entity.
5	**[Defend Entity]**	Defend an entity against attacks.
6	**[Goto Entity]**	Visit an NPC, item or world location
7	**[Use {Item,Skill} on Entity]**	Use an item or skill on a world entity.

Figure 13. Classification of human authored quests by Doran and Parberry (2010)

1) The attack describes all damaging actions performed by the player, for example, kill and shoot
2) Talk refers to conversations between the player and another game character
3) Assemble Item usually requires the player to create a new Item
4) Take-Give-Trade items involve that the layer acquires or drop off an item to someone else
5) Defend means that sometimes the player might have to defend another character from harm or a place from destruction
6) Visit involves the player moving from one location to another
7) Using an acquired item or skill to achieve a task. Tasks that may not be achievable without acquiring the specific skill or task

In standard planning systems, the plans are tailored to cover a variety of these tasks. In order to determine if the quests generated are similar to human-authored quests, I took

200 generated quests; then I classified them based on the criteria above (following Breault, Ouellet & Davies, 2021).

The Classification

To classify the generated quests under the type of human quest they each belong to, I gave each possible action from the list above an identification number. So that each element of a generated plan can be tagged with the identification number of the class it falls under.

Class	ID
Attack	1
Talk	2
Assemble	3
Give-Take-Trade Item	4
Defend	5
Go-to	6
Use	7

Table 4. Action classes and their Identification number

Every element in a generated plan is classified by its actions. For example an element *(retrieve you rifle cultist-base)* of a generated plan gets a classification of *Give-Take-Trade Item class* because the action *retrieve* belongs under a *Give-Take-Trade Item class*.

Likewise an element *(kill you revenant)* in a generated plan gets classified as attack because *kill* is as an attack action. For each generated plan, the class with the most count of actions gets one point. I repeated this process for every plan that was generated by the system. The results of the classification is represented in the table below.

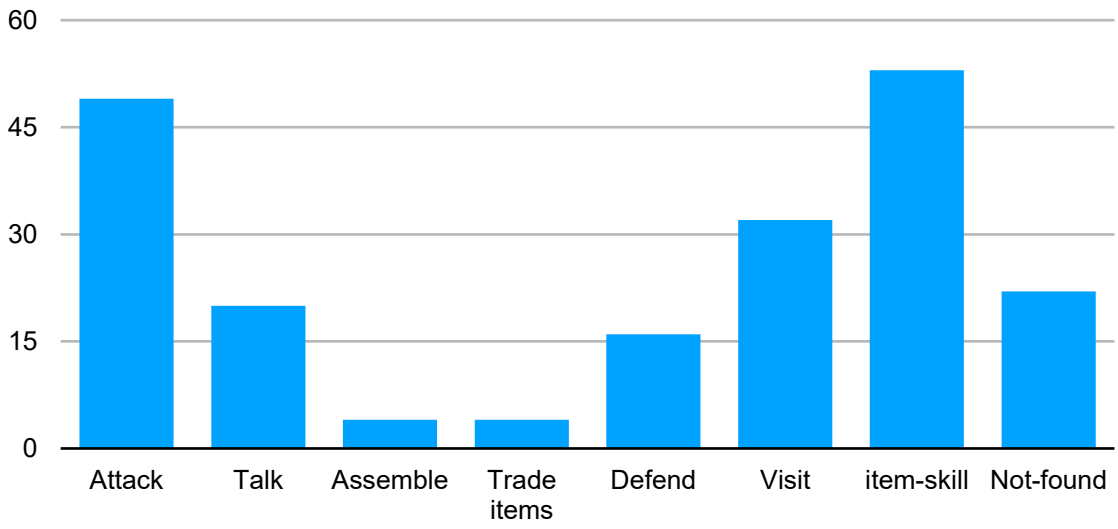

Figure 14. Planning files evaluation in 200 quests

At the end of the classification, 49 quests fell under *attack*, 20 for *talk*, 4 for *trade items*, 16 for *defend,* 32 for *visit* and *assemble* had 4. The highest was *using item or skill* which had 53 quests and lastly, 22 for Not found.

Out of 200 quests only 22 of the quests could not be placed in the categories described by Doran and Parberry (2010), while 178 fit in at least one of the categories. This supports the hypothesis that the plans generated by the system resemble the ones you will find in human-crafted procedural quest planning systems.

Chapter 5

Discussion

A major challenge in evaluating this work is the lack of a large database of stories with labeled segments. To test the event segmentation model extensively, one will need a large database of quest plots with human-labeled boundaries just like in past text segmentation works by Li, Sun & and Joty (2018) as well as Kamath & Wagh (2017). However, because entities in the text (persons vs. locations vs. objects, for example) need to be coded manually, I could not test the segmentation modules on a large corpus of data nor compare the segmentation results with other works on text segmentation.

However, the system's algorithms appear to implement the proposed model. I was also able to evaluate that the plans generated by the system were not only generated correctly, but they are also complete because they execute without breaking and planners find solutions for the files generated by the AQPG system. And for the most important part of the work, I was able to evaluate that the quests generated resemble human-authored quests.

The contents of the quests generated by the system are determined by the contents of the data, if the story is written in a manner that the player character performs quality actions in a clearly described environment, then it will yield quality quests. For example, The *yu-gi-oh's* game plot was used by the AQPG system to generate 24 planning operators while Doom Eternal's story only produced 12 planning operators because yu-gi-oh! contains

more events than in Doom Eternal's story. This means that there will be more possible actions available to the planner to choose from.

There are some limitations to this approach. The problem file does not represent complex relationships between the objects in the world. E.g., knowing that a farmer is on the farm or that a king should be in a castle. These types of relationships might be needed for more complex quest generation. Another limitation is that each action in the plan is independent of any other action in the world. For example, if a certain character has been killed, it should not be available to the planner as the plan proceeds. Also if the player character left a certain location, that location should not be available in the next plan. While this might not affect the generation of side quests where each quest does not depend on an initial quest, it would be helpful and more entertaining if the system is also capable of making quests with actions that are dependent on one another. Also, the preconditions are encoded by picking elements from the same subsegment, the limitation of this is that the system is only able to do type-checking of the objects with this.

The AQPG currently does not fully take advantage of the powers of AI planning such as using specific sequence (layers) of actions to achieve a goal or attaching costs to actions based on some logical rules or efficiently represent complex states. It is also a limitation that the entities at this point are manually encoded. Even though the freedom of being able to allocate labels will make the entity types dynamic, it would be good if the system can correctly identify and label its own entities. An advantage of this feature is that the system will be able to create quests with multiple stories at once without the need to declare entity labels.

In behavioral experiments on EST, events are investigated from each participant's perception of events, that is why it makes sense to use one character at a time in implementing the model. However, in this system, if one changes the name of the player to another character in the story, the system generates a whole new dimension of quests, because in every story, different characters take different actions. For example:

Entities = {

Slayer: Player,

Khan-Maykr: NPC,

Sentinel Prime: Location,

BFG 9000: Weapon

Fortress: Facility,

}

If the above entity data was changed to:

Entities = {

Slayer: NPC,

Khan-Maykr: Player,

Sentinel Prime: Location,

BFG 9000: Weapon

Fortress: Facility,

}

The domain file will be generated to give quests based on the actions of Khan-Maykr in the plot.

This means that if one keeps changing the player label to other entities, one will be able to generate different plans of quests for different characters depending on how the storyline goes. With this, one can generate 100 domain files as long as you have more than 100 entities performing actions in your story. Those 100 plans can also be used to generate thousands of quests given the goals specified the problem files.

Allowing labels for entities is also a good idea because it allows a designer the freedom to change the roles of entities in the world, which will result in a new dimension of quests. Being able to generate plans for different characters is particularly useful in games where the player can choose different characters and explore different paths and experiences.

```
(leave you tea joey kame-game)
(try you atem kame-game tournament)
(add you kame-game millennium-puzzle-box puzzle pelt)
(arrive you tea joey school pelt)
(begin you aigami tournament)
(give you speech)
(practice you aigami girls speech kame-game)
(return you pelt school joey school)
(turn you aigami cards tournament)
; cost = 10 (general cost)
```

```
(kill you gladiator)
(destroy you phobos)
(use you canon)
(find you super-gore-nest)
(travel you satellite-fortress cultist-base)
(inquire you knight)
(retrieve you spellbook)
; cost = 11 (general cost)
```

```
(explore you apex accumulator lazarus office cyberdemon)
(shatter you control-panel argent-supplies panels)
(destroy you apex portal argent-supplies )
(reach you base)
; cost = 11 (general cost)
```

```
(save you tea brother)
(read you bakura news kame-game)
(arrive you tea joey school )
(get you aigami duel)
(begin you aigami tournament)
(give you speech)
; cost = 8 (general cost)
```

```
(save you tea brother)
(read you bakura news kame-game)
(arrive you tea joey school pelt)
(get you aigami duel)
(begin you aigami tournament)
(give you speech)
; cost = 8 (general cost)
```

```
(use you gun zombie)
(kill you revenant)
(destroy you doomblade taras)
(travel you cultist-base)
(inquire you marauder knight)
(retrieve you source)
(return you satellite-fortress)
; cost = 11 (general cost)
```

```
(encounter you sie g22)
(meet you agency america al-samad madison-saint-james )
(learn you agency surkov brayko)
; cost = 8 (general cost)
```

```
(kill you carcas)
(retrieve you crucible)
; cost = 3 (general cost)
```

Figure 15, 16,17,18, 19, 20, 21, 22. Examples of plans generated by the AQPG

Chapter 6

Contributions and Future Direction

6.1 Overview

The major contribution of this project is that I have been able to create a novel architecture for automatically generating plans for procedural narrative generation. In standard game-planning systems, the planning files are carefully crafted by game designers so that stories can be generated dynamically to adapt to game player preferences and increase replay value. For years, that has been a good improvement to the old methods where the whole storyline and sequence of events are manually inputted into game systems. However, generating plans in this way has high costs. It takes too many engineers and time to build these planning systems. Therefore, we must start building AI systems that are creative enough to automatically do these tasks for us.

6.2 AQPG Software

I created a software to implement the architecture that I designed in order to prove my hypothesis that computation of EST can generate planning files for quest generation. The software of this project is available as an open-source project on my GitHub account at github.com/mujsann/AQPG

6.3 Event Segmentation Theory

I have made some contributions to the study of Event Segmentation Theory (EST). I Introduced a computational model of event segmentation for segmenting events in natural language text into discrete events. I explained that the model took into consideration the method in which most quest games unfold (i.e., from location to location). I also added

that bigger events are often a composition of one or more events, represented as a sequence of actions that linearly take place. I subtly made an argument that any computation of Event Segmentation, should be modeled around the actions or perception of one entity given that behavioral studies on Event Segmentation focus on individual perception of events. And because the results of my work showed success in generating quests, I can conclude that computation of event segmentation can be used as a basis for future research in media content generation.

I do not argue that the model which I have designed in this thesis performs even as close to traditional segmentation systems for general purposes. However, if the application of the computation of EST is within the scope of quest segmentation, then the model that I have proposed in this work will serve as a good benchmark for further research.

For future works, an idea birthed from the model that I designed which could not be explored during the course of this work is that event models can also be used to organize plot coherence in a narrative generation. Given that with an event model, humans can understand how the sequence of events unfolds (Zacks et al., 2007) in real-life scenarios, therefore, an agent could also be able to use the model to direct how the sequence of actions should be available to the planning agent. For example, the event model in this work, groups actions based on location, therefore, since each planning operator is a product of a particular location, what future works can explore is how to use location as one of the elements in each planning operator's preconditions. That is, if the current state of the game is not at the location to which the action in question belongs, the action should not be available during planning.

6.4 Procedural Quest generation

I also made some contributions to procedural quest generation. I have argued that procedural content generation is an important area of research. I also argued the present state of procedural generation needs to be improved as the old methods of Procedural Content Generation (PCG) of quests may not be serving some of their intended purposes anymore. Therefore, fresh ideas are needed to move the domain forward. The success of using Event segmentation theory to generate quests as I have demonstrated might inspire more research in EST for the purpose of PCG.

6.5 AI planning

I also made some contributions to the domain of AI planning. I have introduced a novel way of automatically generating planning files. The automatic generation of the Planning files in this work was also done in a domain where it has never been explored before, at least to the best of my knowledge. Also, Past works on Automatic generation of planning files often focus on only one of the planning files (either the domain or problem file), but in this thesis, I have successfully demonstrated how both files needed by the planner can be automatically generated. Additionally, I also showed that the source of generating both files can come from the same data inputs. I have also demonstrated that generating AI planning files may not need a large database nor several hours of data annotation.

Because the results of my work show that computation of Event segmentation was able to successfully encode Artificial Intelligence planning files for quests, my work can therefore also inspire more research in using EST to generate planning files for several other purposes.

The plans generated in this work are not complex. However, they are complete and can be executed as plans to generate quests as I have demonstrated. There are parts of the plans that future works might need to improve on to make this method more useful. For example the preconditions and encoding of states in the domain files. For this method to work in more complex environments and create more interesting quests, the preconditions and states need to be more tightened and improved upon. I have also suggested earlier one way in which this can be achieved.

Multi-agent planning often requires that different agents have their own domain files. By changing the player name in this system, the system will create a domain file that is different from the one of the main characters. Future works can also build up on this idea to explore how it can be used to create collaborative plans for other multi-agent planning systems.

My thesis proposed that the computation of event segmentation theory can automatically generate AI planning files for quest generation. It also provided an architecture for achieving the hypothesis. In my work, I was able to demonstrate from my results that the hypothesis is true. The scope of this project covered contributions in different domains across Cognitive Science and Artificial Intelligence fields. Including but not limited to Procedural Content Generation, AI planning, Natural language Processing and Event Segmentation Theory. Future works can evaluate the quality of the plans in terms of how fun the generated quests are.

References

Alonso, I., & Murillo, A. C. (2019). EV-SegNet: Semantic segmentation for event-based cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* .

Alpha Protocol. Wikipedia, Wikimedia Foundation. (n.d). https://en.wikipedia.org/wiki/Alpha_Protocol

Ammanabrolu, P., Broniec, W., Mueller, A., Paul, J., & Riedl, M. O. (2019). Toward automated quest generation in text-adventure games. *arXiv preprint arXiv:*1909.06283

Anderson, T. S. (2015). From episodic memory to narrative in a cognitive architecture. In *6th Workshop on Computational Models of Narrative (CMN 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Baddeley, A. (2003). Working memory: looking back and looking forward. *Nature reviews neuroscience*, *4*(10), 829-839.

Baguley, T., & Payne, S. J. (2000). Long-term memory for spatial and temporal mental models includes construction processes and model structure. *The Quarterly Journal of Experimental Psychology: Section A*, *53*(2), 479-512.

Barker, R. G., & Wright, H. F. (1954). Midwest and its children: The psychological ecology of an American town.

Barsalou, L. W. (2008). Grounded cognition. *Annu. Rev. Psychol.,* 59, 617-645.

Bates, J. (1994). The role of emotion in believable agents. *Communications of the ACM, 37(7)*, 122-125.

Breault, V., Ouellet, S. & Davies, J. (2021). Let CONAN tell you a story: Procedural quest generation. *Entertainment Computing.* 38, 100422, 1--9.

Boden, M. A. (2004). The creative mind: *Myths and mechanisms.* Psychology Press.

Brants, T., Chen, F., & Tsochantaridis, I. (2002, November). Topic-based document segmentation with probabilistic latent semantic analysis. *In Proceedings of the eleventh international conference on Information and knowledge management* (pp. 211-218)

Brill, E. (1992). *A simple rule-based part of speech tagger*. PENNSYLVANIA UNIV PHILADELPHIA DEPT OF COMPUTER AND INFORMATION SCIENCE.

Caporale, D., Fagiolini, A., Pallottino, L., Settimi, A., Biondo, A., Amerotti, F., ... & Venturini, L. *(2018, September). A planning and control system for self-driving racing vehicles. In 2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI)* (pp. 1-6). IEEE.

Castillo, L., Morales, L., González-Ferrer, A., Fdez-Olivares, J., Borrajo, D., & Onaindía, E. (2010). *Automatic generation of temporal planning domains for e-learning problems. Journal of Scheduling, 13*(4), 347-362.

Cavazza, M., Charles, F., & Mead, S. J. (2001). *narrative representations and causality in character-based interactive storytelling. Proceedings of CAST 2001,* 139-142.

Cullen, J., & Bryman, A. (1988). The knowledge acquisition bottleneck: time for reassessment?. *Expert Systems, 5*(3), 216-225.

Chen, L., Magliano, D. J., & Zimmet, P. Z. (2012). The worldwide epidemiology of type 2 diabetes mellitus—present and future perspectives. *Nature reviews endocrinology, 8*(4), 228.

Chen, Yen-Lin, and Bing-Fei Wu. "A multi-plane approach for text segmentation of complex document images." *Pattern Recognition* 42, no. 7 (2009): 1419-1444.

Doran, J., & Parberry, I. (2010). Towards procedural quest generation: A structural analysis of RPG quests. *Dept. Comput. Sci. Eng., Univ. North Texas, Tech. Rep. LARC-2010*, *2*.

Doom Eternal. fandom (n.d). https://doom.fandom.com/wiki/Doom_Eternal

Elson, D., Dames, N., & McKeown, K. (2010, July). Extracting social networks from literary fiction. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 138-147).

Ferguson, G., & Allen, J. (2007). Mixed-initiative systems for collaborative problem solving. *AI magazine, 28*(2), 23-23

Gärdenfors, P., Jost, J., & Warglien, M. (2018). From actions to effects: Three constraints on event mappings. *Frontiers in psychology,* 9, 1391.

Garrido, A., Morales, L., & Serina, I. (2012). Using AI planning to enhance e-learning processes. In *Proceedings of the International Conference on Automated Planning and Scheduling* (Vol. 22, No. 1).

Griffith, I. (2018). Procedural narrative generation through emotionally interesting non-player characters.

Girju, R., & Moldovan, D. I. (2002). Text mining for causal relations. In *FLAIRS conference* (pp. 360-364).

Yu-Gi-Oh! The Dark Side of Dimensions. fandom. (n.d.). https://yugioh.fandom.com/wiki/Yu-Gi-Oh!_The_Dark_Side_of_Dimensions.

Hard, B. M., Lozano, S. C., & Tversky, B. (2006). Hierarchical encoding of behavior: Translating perception into action. *Journal of Experimental Psychology: General*, *135*(4), 588.

Hearst, M. A. (1994). Multi-paragraph segmentation of expository text. *arXiv preprint cmp-lg/9406037*.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research, 26,* 191-246.

Hendrikx, M., Meijer, S., Van Der Velden, J., & Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 9*(1), 1-22.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780.

Huang, T. H., Ferraro, F., Mostafazadeh, N., Misra, I., Agrawal, A., Devlin, J., ... & Zitnick, C. L. (2016). Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1233-1239).

Jeffrey M Zacks, Nicole K Speer, Khena M Swallow, Todd S Braver, and Jeremy R Reynolds. Event perception: a mind-brain perspective. *Psychological bulletin, 133*(2):273, 2007.

Kamath, S., & Wagh, R. (2017). Named entity recognition approaches and challenges. *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), 6*(2), 259-262.

Kartam, N. A., & Wilkins, D. E. (1990). Towards a foundation for evaluating AI planners. *AI EDAM, 4*(1), 1-13.

Khemlani, S. S., Harrison, A. M., & Trafton, J. G. (2015). Episodes, events, and models. *Frontiers in human neuroscience, 9,* 590.

Kim, J., Banks, C., & Shah, J. (2017). Collaborative planning with encoding of users' high-level strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 31, No. 1).

Korn, O., Blatz, M., Rees, A., Schaal, J., Schwind, V., & Görlich, D. (2017). Procedural content generation for game props? a study on the effects on user experience. *Computers in Entertainment (CIE), 15*(2), 1-15.

Koshorek, O., Cohen, A., Mor, N., Rotman, M., & Berant, J. (2018). Text segmentation as a supervised learning task. *arXiv preprint arXiv:1803.09337*.

Kurby, C. A., & Zacks, J. M. (2012). Starting from scratch and building brick by brick in comprehension. *Memory & cognition*, *40*(5), 812-826.

León, C. (2016). An architecture of narrative memory. *Biologically inspired cognitive architectures, 16,* 19-33.

Li, B., Lee-Urban, S., Appling, D.S., and Riedl, M.O. (2012). Crowdsourcing Narrative Intelligence. *Advances in Cognitive Systems, 2,* 25-42.

Li, J., Sun, A., & Joty, S. R. (2018). SegBot: A Generic Neural Text Segmentation Model with Pointer Network. In *IJCAI* (pp. 4166-4172).

Kozima, H. (1996). Text segmentation based on similarity between words. *arXiv preprint cmp-lg/9601005.*

Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., & Stokes, D. (2004). AI characters and directors for interactive computer games. In *AAAI* (pp. 877-883).

Mateas, M., & Stern, A. (2002). Towards integrating plot and character for interactive drama. In *Socially Intelligent Agents* (pp. 221-228). Springer, Boston, MA.

McKendrick, K. (2017). The Application of Artificial Intelligence in Operations Planning.

Miller, S., & Pennycuff, L. (2008). The power of story: Using storytelling to improve literacy learning. *Journal of Cross-Disciplinary Perspectives in Education, 1*(1), 36-43.

Mitrokhin, A., Ye, C., Fermuller, C., Aloimonos, Y., & Delbruck, T. (2019). EV-IMO: Motion segmentation dataset and learning pipeline for event cameras. *arXiv preprint arXiv:1903.07520.*

Modi, A. (2016). Event embeddings for semantic script modeling. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning* (pp. 75-83).

Mueller, E. T. (2014). *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann.

Newtson, D., & Engquist, G. (1976). The perceptual organization of ongoing behavior. *Journal of Experimental Social Psychology, 12*(5), 436-450.

Original Sin II. Wikipedia, Wikimedia Foundation. (n.d). https://en.wikipedia.org/wiki/Divinity:_Original_Sin_II

Osborn, B. A. (2002). *An agent-based architecture for generating interactive stories*. NAVAL POSTGRADUATE SCHOOL MONTEREY CA.

Porteous, J., Cavazza, M., & Charles, F. (2010). Narrative generation through characters' point of view. In *AAMAS* (pp. 1297-1304).

Rankin, D. M., Scotney, B. W., Morrow, P. J., McDowell, D. R., & Pierscionek, B. K. (2010). Dynamic iris biometry: a technique for enhanced identification. *BMC research notes, 3*(1), 1-7.

Riedl, M. & Young, R. M. (2010). Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39.

Riedl, M., Thue, D., & Bulitko, V. (2011). Game AI as storytelling. In *Artificial intelligence for computer games* (pp. 125-150). Springer, New York, NY.

Riedl, M., & Biemann, C. (2012). Topictiling: a text segmentation algorithm based on lda. In *Proceedings of ACL 2012 Student Research Workshop* (pp. 37-42).

Riedl, M. O. (2016). Computational narrative intelligence: A human-centered goal for artificial intelligence. *arXiv preprint arXiv:1602.06484*.

Sahnoun, S. (2019). Event extraction based on open information extraction and ontology. *arXiv preprint arXiv:1907.00692*.

Schütz-Bosbach, S., & Prinz, W. (2007). Prospective coding in event representation. *Cognitive processing, 8*(2), 93-102.

Shaker, N., Togelius, J., & Nelson, M. J. (2016). *Procedural content generation in games*. Switzerland: Springer International Publishing.

Speer, N. K., Reynolds, J. R., Swallow, K. M., & Zacks, J. M. (2009). Reading stories activates neural representations of visual and motor experiences. *Psychological science, 20*(8), 989-999.

Stefnisson, I., & Thue, D. (2018). Mimisbrunnur: AI-assisted authoring for interactive storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (Vol. 14, No. 1).

Stoffregen, T., Gallego, G., Drummond, T., Kleeman, L., & Scaramuzza, D. (2019). Event-based motion segmentation by motion compensation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 7244-7253)

Sun, Z. Artificial Imagination and Imaginational Intelligence: An Evolutionary Perspective.

Swartjes, I., & Theune, M. (2008). The virtual storyteller: Story generation by simulation. In *Proceedings of the 20th Belgian-Netherlands Conference on Artificial Intelligence (BNAIC)* (pp. 257-264).

Thue, D., Schiffel, S., Árnason, R. A., Stefnisson, I. S., & Steinarsson, B. (2016). Delayed roles with authorable continuity in plan-based interactive storytelling. In *International Conference on Interactive Digital Storytelling* (pp. 258-269). Springer, Cham.

Togelius, J., Champandard, A. J., Lanzi, P. L., Mateas, M., Paiva, A., Preuss, M., & Stanley, K. O. (2013). Procedural content generation: Goals, challenges and actionable steps. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games, 3*(3), 172-186.

Turner, S. R. (2014). The creative process: A computer model of storytelling and creativity. Psychology Press.

Utiyama, M., & Isahara, H. (2001). A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (pp. 499-506).

Vo, D. T., & Bagheri, E. (2018). Open information extraction. In *Semantic Computing* (pp. 3-8).

Winston, P.H. (2011). The Strong Story Hypothesis and the Directed Perception Hypothesis.

Wan, J., O'grady, M. J., & O'Hare, G. M. (2015). Dynamic sensor event segmentation for real-time activity recognition in a smart home context. *Personal and Ubiquitous Computing, 19*(2), 287-301.

Wiessner. W Polly 2014. Embers of society: Firelight talk among the ju/hoansi bushmen. *Proceedings of the National Academy of Sciences, 111*(39):14027–14035.

Young, R. M., Ware, S. G., Cassell, B. A., & Robertson, J. (2013). Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative*, 37(1-2), 41-64.

Zacks, J. M., Speer, N. K., Swallow, K. M., Braver, T. S., & Reynolds, J. R. (2007). Event perception: a mind-brain perspective. *Psychological bulletin*, 133(2), 273.

Zacks, J. M., Speer, N. K., Swallow, K. M., & Maley, C. J. (2010). The brain's cutting-room floor: Segmentation of narrative cinema. *Frontiers in human neuroscience*, *4*, 168.